In the Claims:

Please amend claims 1-2, 5, 8, 10-12, 15, 18, 20-22, 28, 30-32, 34, 38-39, 41, 45-46, and 48. Please add new claims 52-57. The claims are as follows:

1. (Currently amended) A method of inserting ~~one or more~~ a plurality of global breakpoints into computer software for debugging the computer software, said method performed by executing a computer program on a processor of a computer system, said executing including performing the steps of:

~~inserting a global~~ breakpoint ~~in a page containing software code if said page is present in memory;~~

~~reading said page into memory if not present in memory, and inserting a global breakpoint in said page immediately after being read into memory; and~~

~~detecting a private copy of said page if present, and inserting a global breakpoint in said private copy~~

performing a first loop over the global breakpoints such that for each global breakpoint in the first loop: determining the page into which the global breakpoint is to be inserted, reading the page into memory if the page is not present in memory, and inserting the global breakpoint in the page present in memory; and

performing a second loop over the global breakpoints such that for each global breakpoint in the second loop: determining if a private copy exists for the page into which the global breakpoint is to be inserted, detecting the private copy if the private copy exists, reading the private copy into memory if the private copy exists and is not present in memory, and inserting

the global breakpoint in the private copy present in memory.

2. (Currently amended) The method according to claim 1, wherein said reading step includes the step of providing a readpage process for reading said page into memory and for inserting [[a]] the global breakpoint in said page immediately after being read into memory.

3. (Original) The method according to claim 2, wherein said readpage process is implemented as a kernel routine that is called when said page is loaded into memory.

4. (Original) The method according to claim 2, wherein said reading step includes the step of setting up an operation to insert said global breakpoint in said page immediately after said page is read into memory by an operating system, if said page is not already in memory.

5. (Currently amended) The method according to claim 4, wherein said readpage process [[is]] comprises changing a file specific readpage process to a wrapper routine that invokes an original readpage process and then performs said operation required.

6. (Original) The method according to claim 1, wherein said detecting step includes the step of swapping said copy to a swap device after inserting said global breakpoint in said copy.

7. (Original) The method according to claim 6, wherein said detecting step includes the further step of marking said copy as dirty after inserting said global breakpoint in said copy, whereby

when swapping said copy to said swap device, said global breakpoint being present in said swapped copy.

8. (Currently amended) The method according to claim 1, said executing further including the step of identifying ~~said~~ each global breakpoint using an identifier of a file and an offset in said file.

9. (Original) The method according to claim 8, wherein said file identifier is an inode.

10. (Currently amended) The method according to claim 8, further including during execution of said first loop the step of determining if said page is present in memory using a lookup table based on said file identifier and said offset associated with said global breakpoint.

11. (Currently amended) A computer-implemented apparatus for inserting ~~one or more~~ a plurality of global breakpoints into computer software for debugging the computer software, said apparatus including:

a central processing unit for executing said computer software;

memory for storing at least a portion of said computer software;

~~means for inserting a global breakpoint in a page containing software code if said page is present in memory;~~

~~means for reading said page into memory if not present in said memory, and inserting a global breakpoint in said page immediately after being read into memory; and~~

~~means for detecting a private copy of said page if present, and inserting a global breakpoint in said private copy~~

first means for performing a first loop over the global breakpoints such that for each global breakpoint in the first loop said first means includes: means for determining the page into which the global breakpoint is to be inserted, means for reading the page into memory if the page is not present in memory, and means for inserting the global breakpoint in the page present in memory; and

second means for performing a second loop over the global breakpoints such that for each global breakpoint in the second loop said second means includes: means for determining if a private copy exists for the page into which the global breakpoint is to be inserted, means for detecting the private copy if the private copy exists, means for reading the private copy into memory if the private copy exists and is not present in memory, and means for inserting the global breakpoint in the private copy present in memory.

12. (Currently amended) The apparatus according to claim 11, wherein said reading means includes means for providing a readpage process for reading said page into said memory and being adapted to insert [[a]] the global breakpoint in said page immediately after being read into memory.

13. (Original) The apparatus according to claim 12, wherein said readpage process is implemented as a kernel routine that is called when said page is loaded into said memory.

14. (Original) The apparatus according to claim 11, wherein said reading means includes means for setting up an operation to insert said global breakpoint in said page immediately after said page is read into memory by an operating system, if said page is not already in memory.

15. (Currently amended) The apparatus according to claim 14, wherein said readpage process ~~is adapted by~~ comprises changing a file specific readpage process to a wrapper routine that invokes an original readpage process and then performs the operation required.

16. (Original) The apparatus according to claim 11, wherein said detecting means includes means for swapping said copy to a swap device after inserting said global breakpoint in said copy.

17. (Original) The apparatus according to claim 16, wherein said detecting means includes means for marking said copy as dirty after inserting said global breakpoint in said copy, whereby when swapping said copy to a swap device, said global breakpoint being present in said swapped copy.

18. (Currently amended) The apparatus according to claim 11, further including means for identifying ~~said~~ each global breakpoint using an identifier of a file and an offset in said file.

19. (Original) The apparatus according to claim 18, wherein said file identifier is an inode.

20. (Currently amended) The apparatus according to claim 18, further including means for determining during execution of said first loop if said page is present in said memory using a

lookup table based on said file identifier and said offset associated with said global breakpoint.

21. (Currently amended) A computer program product having a computer readable medium having a computer program recorded therein for inserting ~~one or more~~ a plurality of global breakpoints into computer software for debugging the computer software, said computer program product including:

~~computer program code means for inserting a global breakpoint in a page containing software code if said page is present in memory;~~

~~computer program code means for reading said page into memory if not present in said memory, and inserting a global breakpoint in said page immediately after being read into memory; and~~

~~computer program code means for detecting a private copy of said page if present, and inserting a global breakpoint in said private copy~~

first computer program code means for performing a first loop over the global breakpoints such that for each global breakpoint in the first loop said first means includes: computer program code means for determining the page into which the global breakpoint is to be inserted, computer program code means for reading the page into memory if the page is not present in memory, and computer program code means for inserting the global breakpoint in the page present in memory; and

second computer program code means for performing a second loop over the global breakpoints such that for each global breakpoint in the second loop said second means includes: computer program code means for determining if a private copy exists for the page into which

the global breakpoint is to be inserted, computer program code means for detecting the private copy if the private copy exists, computer program code means for reading the private copy into memory if the private copy exists and is not present in memory, and computer program code means for inserting the global breakpoint in the private copy present in memory.

22. (Currently amended) The computer program product according to claim 21, wherein said computer program code means for reading includes computer program code means for providing a readpage process for reading said page into said memory and for inserting [[a]] the global breakpoint in said page immediately after being read into memory.

23. (Original) The computer program product according to claim 22, wherein said readpage process is implemented as a kernel routine that is called when said page is loaded into said memory.

24. (Original) The computer program product according to claim 22, wherein said computer program code means for reading includes computer program code means for setting up an operation to insert said global breakpoint in said page immediately after said page is read into memory by an operating system, if said page is not already in memory.

25. (Previously presented) The computer program product according to claim 22, wherein said readpage process comprises changing a file specific readpage process to a wrapper routine that invokes an original readpage process and then performs said operation required.

26. (Original) The computer program product according to claim 21, wherein said computer program code means for detecting includes computer program code means for swapping said copy to a swap device after inserting said global breakpoint in said copy.

27. (Original) The computer program product according to claim 26, wherein said computer program code means for detecting includes computer program code means for marking said copy as dirty after inserting said global breakpoint in said copy, whereby when swapping said copy to a swap device, said global breakpoint being present in said swapped copy.

28. (Currently amended) The computer program product according to claim 21, further including computer program code means for identifying said each global breakpoint using an identifier of a file and an offset in said file.

29. (Original) The computer program product according to claim 28, wherein said file identifier is an inode.

30. (Currently amended) The computer program product according to claim 28, further including computer program code means for determining during execution of said first loop if said page is present in said memory using a lookup table based on said file identifier and said offset associated with said global breakpoint.

31. (Currently amended) A method of removing ~~one or more~~ a plurality of global breakpoints for

~~debugging~~ from computer software, said method performed by executing a computer program on a processor of a computer system, said executing including performing the steps of:

~~removing a global breakpoint in a page containing software code if said page containing said global breakpoint is present in memory; and~~

~~detecting a private copy of said page if present, reading said page into memory if not present in memory, and removing a global breakpoint in said private copy~~

performing a first loop over the global breakpoints such that for each global breakpoint in the first loop: determining the page from which the global breakpoint is to be removed, reading the page into memory if the page is not present in memory, and removing the global breakpoint from the page present in memory; and

performing a second loop over the global breakpoints such that for each global breakpoint in the second loop: determining if a private copy exists for the page from which the global breakpoint is to be removed, detecting the private copy if the private copy exists, reading the private copy into memory if the private copy exists and is not present in memory, and removing the global breakpoint from the private copy present in memory.

32. (Currently amended) The method according to claim 31, wherein said reading step includes the step of providing a readpage process for reading said page into memory and for removing [[a]] the global breakpoint in said page immediately after being read into memory.

33. (Original) The method according to claim 32, wherein said readpage process is implemented as a kernel routine that is called when said page is loaded into memory.

34. (Currently amended) The method according to claim 31, wherein said reading step includes the step of turning off an operation set up earlier for inserting [[a]] the global breakpoint in said page when said page is read into memory.

35. (Original) The method according to claim 31, further including the step of identifying said global breakpoint using an identifier of a file and an offset in said file.

36. (Original) The method according to claim 35, wherein said file identifier is an inode.

37. (Original) The method according to claim 35, further including the step of determining if said page is present in memory using a lookup table based on said file identifier and said offset.

38. (Currently amended) A computer-implemented apparatus for removing ~~one or more~~ a plurality of global breakpoints ~~for debugging~~ from computer software, said apparatus including:

a central processing unit for executing said computer software;

memory for storing at least a portion of said computer software;

~~means for removing a global breakpoint in a page containing software code if said page containing said global breakpoint is present in memory; and~~

~~means for detecting a private copy of said page if present, reading said page into memory if not present in said memory, and removing a global breakpoint in said private copy~~

first means for performing a first loop over the global breakpoints such that for each global breakpoint in the first loop: means for determining the page from which the global

breakpoint is to be removed, means for reading the page into memory if the page is not present in memory, and means for removing the global breakpoint from the page present in memory; and

second means for performing a second loop over the global breakpoints such that for each global breakpoint in the second loop: means for determining if a private copy exists for the page from which the global breakpoint is to be removed, means for detecting the private copy if the private copy exists, means for reading the private copy into memory if the private copy exists and is not present in memory, and means for removing the global breakpoint from the private copy present in memory.

39. (Currently amended) The apparatus according to claim 38, wherein said reading means includes means for providing a readpage process for reading said page into said memory and for removing [[a]] said global breakpoint in said page immediately after being read into memory.

40. (Original) The apparatus according to claim 39, wherein said readpage process is implemented as a kernel routine that is called when said page is loaded into said memory.

41. (Currently amended) The apparatus according to claim 38, wherein said reading means includes means for turning off an operation set up earlier for inserting [[a]] the global breakpoint in said page when said page is read into memory.

42. (Original) The apparatus according to claim 38, further including means for identifying said global breakpoint using an identifier of a file and an offset in said file.

43. (Original) The apparatus according to claim 42, wherein said file identifier is an inode.

44. (Original) The apparatus according to claim 42, further including means for determining if said page is present in said memory using a lookup table based on said file identifier and said offset.

45. (Currently amended) A computer program product having a computer readable medium having a computer program recorded therein for removing ~~one or more~~ a plurality of global breakpoints ~~for debugging~~ from computer software, said computer program product including:

~~computer program code means for removing a global breakpoint in a page containing software code if said page containing said global breakpoint is present in memory; and~~

~~computer program code means for detecting a private copy of said page if present, reading said page into memory if not present in said memory, and removing a global breakpoint in said private copy~~

first computer program code means for performing a first loop over the global breakpoints such that for each global breakpoint in the first loop: computer program code means for determining the page from which the global breakpoint is to be removed, computer program code means for reading the page into memory if the page is not present in memory, and computer program code means for removing the global breakpoint from the page present in memory; and

second computer program code means for performing a second loop over the global breakpoints such that for each global breakpoint in the second loop: computer program code means for determining if a private copy exists for the page from which the global breakpoint is to

be removed, computer program code means for detecting the private copy if the private copy exists, computer program code means for reading the private copy into memory if the private copy exists and is not present in memory, and computer program code means for removing the global breakpoint from the private copy present in memory.

46. (Currently amended) The computer program product according to claim 45, wherein said computer program code means for reading includes computer program code means for providing a readpage process for reading said page into said memory and for removing [[a]] the global breakpoint in said page immediately after being read into memory.

47. (Original) The computer program product according to claim 45, wherein said readpage process is implemented as a kernel routine that is called when said page is loaded into said memory.

48. (Currently amended) The computer program product according to claim 46, wherein said computer program code means for reading includes turning off an operation set up earlier for inserting [[a]] the global breakpoint in said page computer program code means for when said page is read into memory.

49. (Original) The computer program product according to claim 45, further including computer program code means for identifying said global breakpoint using an identifier of a file and an offset in said file.

50. (Original) The computer program product according to claim 49, wherein said file identifier is an inode.

51. (Original) The computer program product according to claim 49, further including computer program code means for determining if said page is present in said memory using a lookup table based on said file identifier and said offset.

52. (New) The method according to claim 1, wherein the second loop is performed after the first loop is performed.

53. (New) The apparatus according to claim 11, further comprising means for performing the second loop after the first loop is performed.

54. (New) The computer program product according to claim 21, further comprising computer program code means for performing the second loop after the first loop is performed.

55. (New) The method according to claim 31, wherein the second loop is performed after the first loop is performed.

56. (New) The apparatus according to claim 38, further comprising means for performing the second loop after the first loop is performed.

57. (New) The computer program product according to claim 45, further comprising computer program code means for performing the second loop after the first loop is performed.